ARTIFICIAL INTELLIGENCE ANALYTICS

SEMESTER PROJECT

SUBMITTED BY

FIZZA KHAN (2021-BSE-012)

LAIBA ARIF (2021-BSE-015)

TANZEELA ASGHAR(2021-BSE-032)

LARAIB IJAZ (2021-BSE-036)

SEMESTER/SECTION

V-A

SUBMITTED TO

Dr. Irum Matloob

SUBMITTED ON

17th JAN 2024

DATASET:

CUSTOMER SHOPPING TRENDS

https://www.kaggle.com/datasets/iamsouravbanerjee/customer-shopping-trends-dataset

CONCERNS OF THIS DATASET:

The dataset on Customer Shopping Preferences provides crucial insights into consumer habits and buying trends, offering businesses a valuable tool for tailoring their products, marketing approaches, and overall customer interactions. By delving into this data, businesses can gain valuable information to make well-informed decisions, refine their product offerings, and elevate overall customer satisfaction. This dataset serves as a strategic resource for companies seeking to align their strategies with the ever-evolving needs and preferences of their customers.

1: DATA PREPROCESSING

Importing Libraries: Start by importing the necessary libraries, such as NumPy, Pandas,and scikit-learn, which are commonly used for data preprocessing tasks.

```
#DATA PREPROCESSING
#Importing Libraries
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler
```

Loading Data: Read your dataset into a Pandas Data Frame. You can use Pandas to read data from various sources like CSV, Excel, SQL databases, and more.

```
#Loading Data
df1 = pd.read_csv('shopping_trends.csv')
```

Handling Missing Data: Deal with missing values by removing rows with missing data, filling in missing values with the mean or median, or using more advanced imputation techniques.

```
#Fill missing values with the mean
df1.fillna(df1.mean(), inplace=True)
```

Handling Categorical Data: Convert categorical variables into numerical representations using techniques like one-hot encoding or label encoding.

 # One-hot encoding:

```
#One hot encoding
print(df1.columns)
df1=pd.get_dummies(df1, columns=['Gender'])
df1=pd.get_dummies(df1, columns=['Size'])
df1=pd.get_dummies(df1, columns=['Subscription Status'])
df1=pd.get_dummies(df1, columns=['Discount Applied'])
df1=pd.get_dummies(df1, columns=['Promo Code Used'])
```

 # Label encoding:

```
#Label Encoding
label_encoder = LabelEncoder()
df1['Item Purchased'] = label_encoder.fit_transform(df1['Item Purchased'])
df1['Category'] = label_encoder.fit_transform(df1['Category'])
df1['Location'] = label_encoder.fit_transform(df1['Location'])
df1['Color'] = label_encoder.fit_transform(df1['Color'])
df1['Season'] = label_encoder.fit_transform(df1['Season'])
df1['Payment Method'] = label_encoder.fit_transform(df1['Payment Method'])
df1['Shipping Type'] = label_encoder.fit_transform(df1['Shipping Type'])
df1['Preferred Payment Method'] = label_encoder.fit_transform(df1['Preferred Payment Method'])
df1['Frequency of Purchases'] = label_encoder.fit_transform(df1['Frequency of Purchases'])
```

Feature Scaling: Standardize or normalize numerical features to ensure they have the same scale, which can improve the performance of many machine learning algorithms.

```
#Feature Scaling
scaler = StandardScaler()
df1[['Customer ID', 'Age' , 'Purchase Amount (USD)' , 'Review Rating' ,
'Previous Purchases']] = scaler.fit_transform(df1[['Customer ID', 'Age' ,
'Purchase Amount (USD)' , 'Review Rating' , 'Previous Purchases']])
```

2: FEATURE SELECTION:

Feature Selection can be done through Genetic algorithm and Decision tree Algorithm. We apply both Algorithms on our dataset. Decision tree Algorithm selects four features whereas Genetic Algorithm selects eight features. So we go for genetic algorithm.

GENETIC ALGORITHM

```
#FEATURE SELECTION
#GENETIC ALGORITHM
import random
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
# Load the dataset
df=pd.read_csv('shopping_trends.csv')
```

```
# Extract features and target from your dataset
X = df1[['Age', 'Item Purchased', 'Category', 'Location', 'Color',
'Season', 'Payment Method', 'Shipping Type', 'Preferred Payment
Method','Review Rating', 'Purchase Amount (USD)' , 'Previous Purchases']]
y = df1[['Review Rating']]
```

```
# Step 1: Choose an encoding technique, a selection operator, and a crossover operator
def encode_chromosome(num_features):
    return [random.randint(0, 1) for _ in range(num_features)]

def select_parents(population, fitness_scores):
    return random.choices(population, weights=fitness_scores, k=2)

def crossover(parents, crossover_rate):
    child1, child2 = parents
    if random.random() < crossover_rate:
        crossover_point = random.randint(1, len(child1) - 1)
        child1[crossover_point:], child2[crossover_point:] = child2[crossover_point:], child1[crossover_point:]
    return child1, child2

    # Mutation function
def mutate(chromosome, mutation_rate):
    mutated_chromosome = chromosome.copy()
    for i in range(len(mutated_chromosome)):
        if random.random() < mutation_rate:
            # Flip the bit at position i
            mutated_chromosome[i] = 1 - mutated_chromosome[i]
    return mutated_chromosome
```

```python
# Step 2: Choose a population size
POPULATION_SIZE = 10

# Step 3: Randomly choose the initial population
def generate_initial_population(num_features, population_size):
    return [encode_chromosome(num_features) for _ in range(population_size)]

# Step 4: Select parental chromosomes
def evaluate_population(population, X_train, y_train, num_features):
    fitness_scores = []
    for chromosome in population:
        fitness_scores.append(sum(chromosome))
    return fitness_scores

# Step 5: Perform Crossover and Mutation
def generate_offsprings(population, fitness_scores, crossover_rate, mutation_rate):
    offsprings = []
    while len(offsprings) < len(population):
        parents = select_parents(population, fitness_scores)
        child1, child2 = crossover(parents, crossover_rate)

        # Apply mutation to the offspring
        child1 = mutate(child1, mutation_rate)
        child2 = mutate(child2, mutation_rate)

        offsprings.append(child1)
        offsprings.append(child2)
    return offsprings

# Step 6: Evaluation of offsprings
def evaluate_population(population, X_train, y_train, num_features):
    fitness_scores = []
    for chromosome in population:
        # Count the number of ones in the chromosome (selected features)
        num_ones = sum(chromosome)
        fitness_scores.append(num_ones)
    return fitness_scores

# Select survivors
def select_survivors(population, offsprings, fitness_scores):
    population_with_offsprings = population + offsprings
    sorted_indices = np.argsort(fitness_scores)[::-1]
    return [population_with_offsprings[i] for i in sorted_indices[:len(population)]]
```

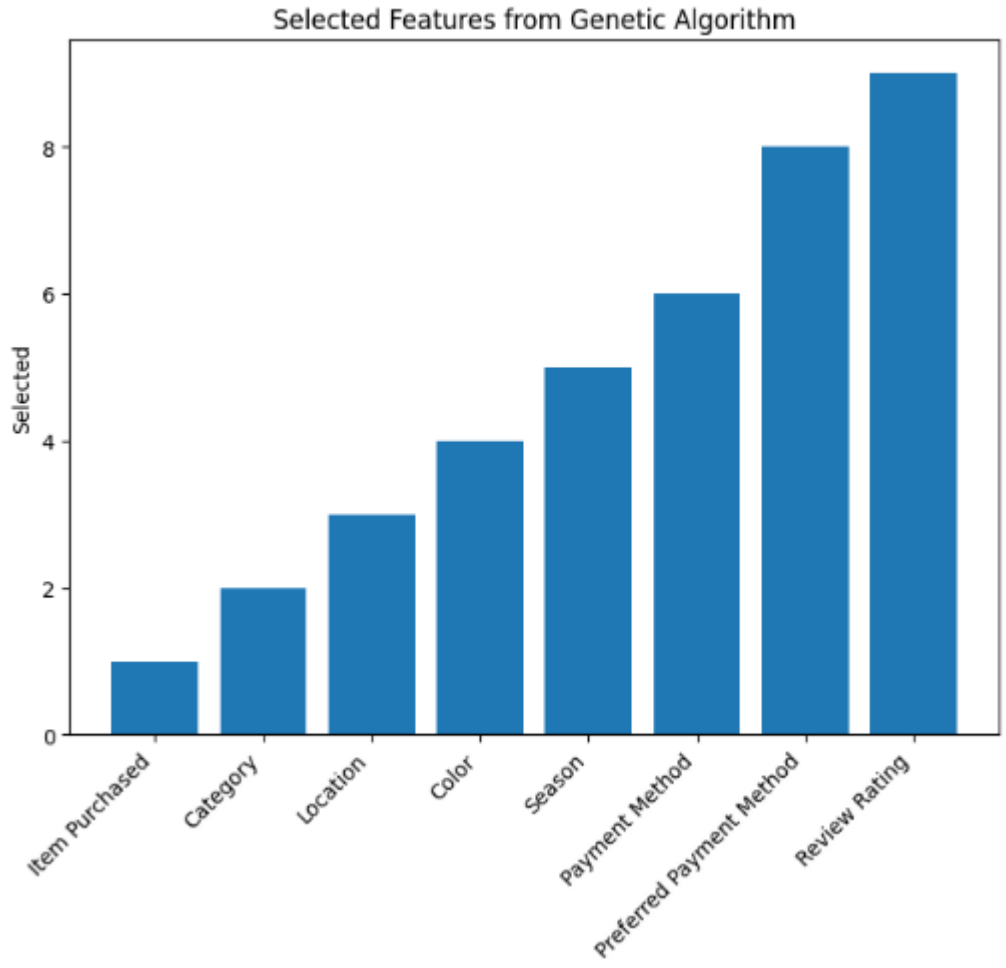
```python
# Step 7: Repeat the process and the fitness function should be maxone where the fitness of a chromosome is determined by the number of ones in it
def genetic_algorithm(X_train, y_train, num_features, num_generations, crossover_rate,mutation_rate):
    population = generate_initial_population(num_features, POPULATION_SIZE)
    for _ in range(num_generations):
        fitness_scores = evaluate_population(population, X_train, y_train, num_features)
        offsprings = generate_offsprings(population, fitness_scores, crossover_rate,mutation_rate)
        population = select_survivors(population, offsprings, fitness_scores)
    best_chromosome = population[0]
    selected_features = [i for i, bit in enumerate(best_chromosome) if bit == 1]
    return selected_features

# Example usage with mutation rate
num_generations = 10
crossover_rate = 0.8
mutation_rate = 0.01  # Adjust the mutation rate as needed
selected_features = genetic_algorithm(X, y, len(X.columns), num_generations, crossover_rate, mutation_rate)

# Print the selected features
print("Selected Features:", selected_features)
# Plotting the selected features
plt.figure(figsize=(8, 6))
plt.bar(X.columns[selected_features], selected_features)  # Use only the selected features
```



```python
plt.xlabel('Features')
plt.ylabel('Selected')
plt.title('Selected Features from Genetic Algorithm')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better visibility
plt.show()
```

OUTPUT:

```
Selected Features: [1, 2, 3, 4, 5, 6, 8, 9]
```



ANALYSIS:

This code implements a genetic algorithm for feature selection. It begins by loading a dataset named 'shopping_trends.csv' and extracting features (X) and the target variable (y). The genetic algorithm follows standard steps: Chromosome encoding, Population initialization, Parent selection, Crossover, Evaluation, and Survivor selection. Chromosomes are binary strings representing feature presence or absence. The initial population is generated randomly, and parents are selected based on fitness scores. Crossover occurs with a specified probability, creating offspring chromosomes. The evaluation function calculates fitness based on the number of selected features. Survivor selection combines the current population and offspring, retaining the fittest individuals. The algorithm is iteratively executed for a defined number of generations and a given crossover rate

CLUSTERING:

The Customer Trends Shopping dataset contains unsupervised data because it is not labeled data and doesn't contain Decision attribute in it.

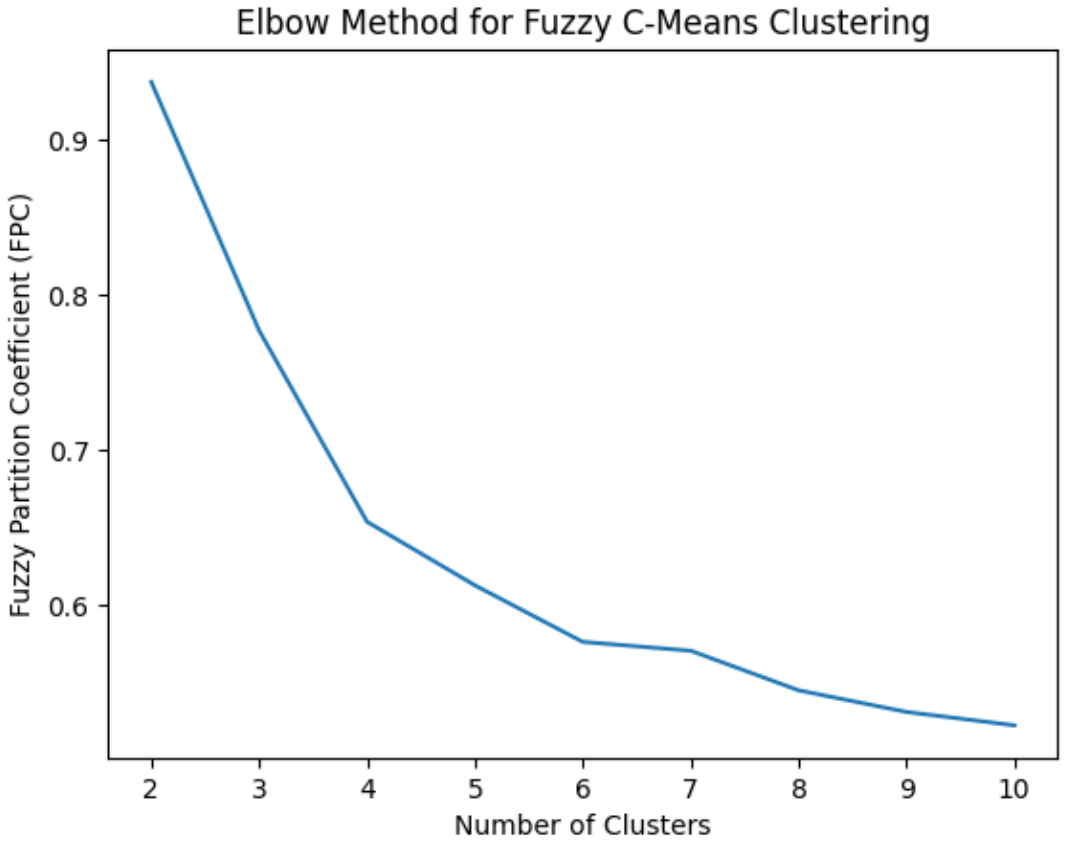
That's why we apply clustering on our dataset. Through **elbow method** we got four optimal clusters for our dataset and we use **Fuzzy C Means clustering algorithm**. The algorithm involves initializing cluster centers and a membership matrix, where each entry represents the degree of membership of a data point to a particular cluster. Iteratively, the membership matrix and cluster centers are updated based on a fuzzy membership function, and the process continues until convergence is achieved. The fuzziness of the assignments is controlled by a parameter called the fuzziness parameter (m)

```
#CLUSTERING
#FFUZZY C MEANS ALGORITHM
# Import necessary libraries
!pip install -U scikit-fuzzy
import numpy as np
import pandas as pd
import skfuzzy as fuzz
from matplotlib import pyplot as plt

# Generate sample data
np.random.seed(42)
df1 = np.vstack((np.random.normal(0, 1, (100, 2)),
                 np.random.normal(5, 1, (100, 2))))


# Fuzzy C-Means clustering
def fuzzy_cmeans(df1, num_clusters, m=2, max_iter=100, error=0.005):
    cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(
        df1.T, num_clusters, m, error, max_iter, init=None)
    return cntr, u, u0, d, jm, p, fpc
```

```python
# Elbow method to determine the optimal number of clusters
def elbow_method(df1, max_clusters=10):
    fpcs = []
    for n_clusters in range(2, max_clusters + 1):
        cntr, u, _, _, _, _, fpc = fuzzy_cmeans(df1, n_clusters)
        fpcs.append(fpc)
    return fpcs

# Plot the elbow method graph
def plot_elbow(fpcs):
    fig, ax = plt.subplots()
    ax.plot(range(2, len(fpcs) + 2), fpcs)
    ax.set_xlabel('Number of Clusters')
    ax.set_ylabel('Fuzzy Partition Coefficient (FPC)')
    ax.set_title('Elbow Method for Fuzzy C-Means Clustering')
    plt.show()
```

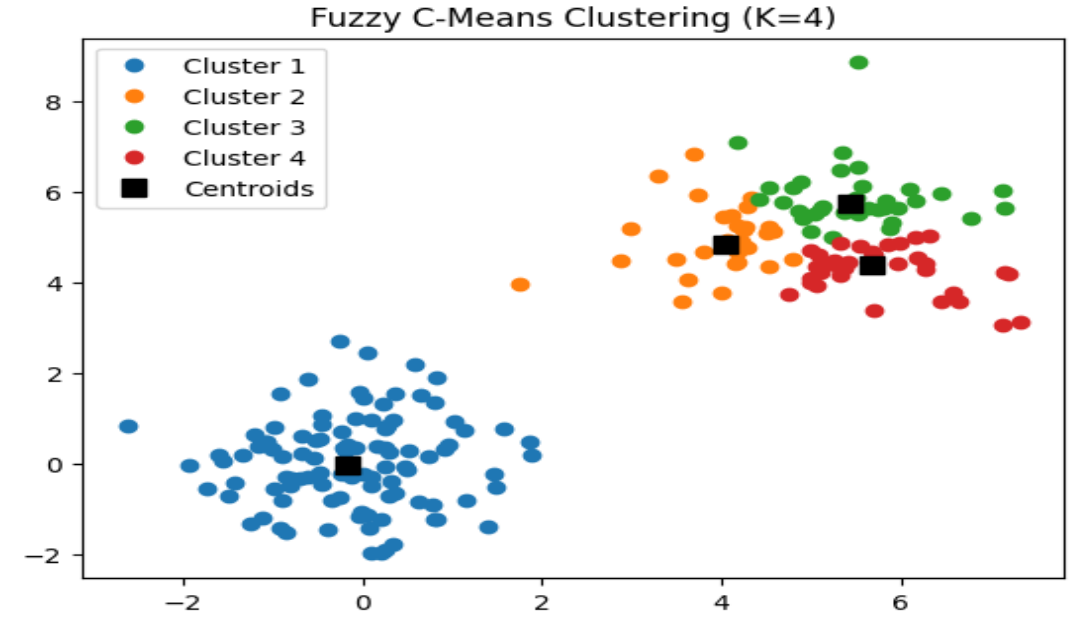
```python
# Example usage
max_clusters = 10
fpcs = elbow_method(df1, max_clusters)
plot_elbow(fpcs)

# Choose the optimal number of clusters based on the elbow point
optimal_num_clusters = np.argmax(fpcs) + 4
print(f'Optimal number of clusters: {optimal_num_clusters}')

# Perform Fuzzy C-Means clustering with the optimal number of clusters
cntr, u, _, _, _, _, _ = fuzzy_cmeans(df1, optimal_num_clusters)

# Plot the clustered data
fig, ax = plt.subplots()
cluster_membership = np.argmax(u, axis=0)
for i in range(optimal_num_clusters):
    ax.plot(df1[cluster_membership == i, 0], df1[cluster_membership == i, 1], 'o', label=f'Cluster {i + 1}')

ax.plot(cntr[:, 0], cntr[:, 1], 'ks', markersize=8, label='Centroids')
ax.legend()
ax.set_title(f'Fuzzy C-Means Clustering (K={optimal_num_clusters})')
plt.show()
```

OUTPUT:




```
Optimal number of clusters: 4
```

```
Optimal number of clusters: 4
```



ANALYSIS:

The Fuzzy C-Means clustering algorithm is encapsulated in the fuzzy_cmeans function, which computes cluster centers, membership matrices, and other relevant parameters. The script then employs the elbow method, implemented in the elbow_method function, to determine the optimal number of clusters by evaluating the Fuzzy Partition Coefficient (FPC) for different cluster numbers. The resulting FPC values are plotted using the plot_elbow function, facilitating the identification of the elbow point. The optimal number of clusters is automatically selected, and Fuzzy C-Means clustering is performed with this number. Finally, the script visualizes the clustered data points and cluster centroids.

4: PERFORMANCE MATRIX:

The silhouette score measures how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The score ranges from -1 to 1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.

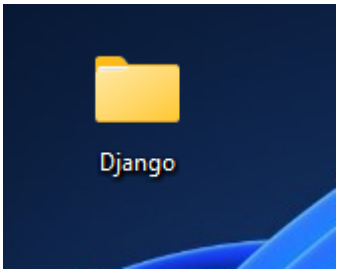
```
#PERFORMANCE MATRIX
!pip install -U scikit-fuzzy

import numpy as np
from sklearn.datasets import make_blobs
import skfuzzy
from sklearn.metrics import silhouette_score

# Generate some sample data (replace this with your own unsupervised data)
df1, _ = make_blobs(n_samples=300, centers=3, cluster_std=0.60, random_state=42)

# Fuzzy C-means clustering
n_clusters = 4  # Set the number of clusters
fuzzy_centers, u, _, _, _, _, _ = skfuzzy.cmeans(df1.T, n_clusters, 2, error=0.005, maxiter=1000, init=None)

# Calculate the fuzzy labels
labels = np.argmax(u, axis=0)

# Calculate the silhouette score
silhouette_avg = silhouette_score(df1, labels)
print(f"Silhouette Score: {silhouette_avg}")
```

RESULT:

```
Silhouette Score: 0.7361389221412058
```

5: FRONT USING DJANGO:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Laraib Ijaz\Desktop> pip install django
Collecting django
  Obtaining dependency information for django from https://files.pythonhosted.org/packages/97/67/6804ff6fc
8ddc2d0a500963b0801a97b7ec08a/Django-5.0.1-py3-none-any.whl.metadata
  Downloading Django-5.0.1-py3-none-any.whl.metadata (4.2 kB)
Collecting asgiref<4,>=3.7.0 (from django)
  Obtaining dependency information for asgiref<4,>=3.7.0 from https://files.pythonhosted.org/packages/9b/8
d8ffc89232711b4e618c15cb7a392a17384bbeef/asgiref-3.7.2-py3-none-any.whl.metadata
  Downloading asgiref-3.7.2-py3-none-any.whl.metadata (9.2 kB)
Collecting sqlparse>=0.3.1 (from django)
  Downloading sqlparse-0.4.4-py3-none-any.whl (41 kB)
     ---------------------------------------- 41.2/41.2 kB 76.2 kB/s eta 0:00:00
Collecting tzdata (from django)
  Obtaining dependency information for tzdata from https://files.pythonhosted.org/packages/a3/fb/52b62131e
29647dad0e0051a92bb66b43c70ff/tzdata-2023.4-py2.py3-none-any.whl.metadata
  Downloading tzdata-2023.4-py2.py3-none-any.whl.metadata (1.4 kB)
Downloading Django-5.0.1-py3-none-any.whl (8.1 MB)
     ---------------------------------------- 8.1/8.1 MB 638.2 kB/s eta 0:00:00
Downloading asgiref-3.7.2-py3-none-any.whl (24 kB)
Downloading tzdata-2023.4-py2.py3-none-any.whl (346 kB)
     ---------------------------------------- 346.6/346.6 kB 2.4 MB/s eta 0:00:00
Installing collected packages: tzdata, sqlparse, asgiref, django
Successfully installed asgiref-3.7.2 django-5.0.1 sqlparse-0.4.4 tzdata-2023.4

[notice] A new release of pip is available: 23.2.1 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

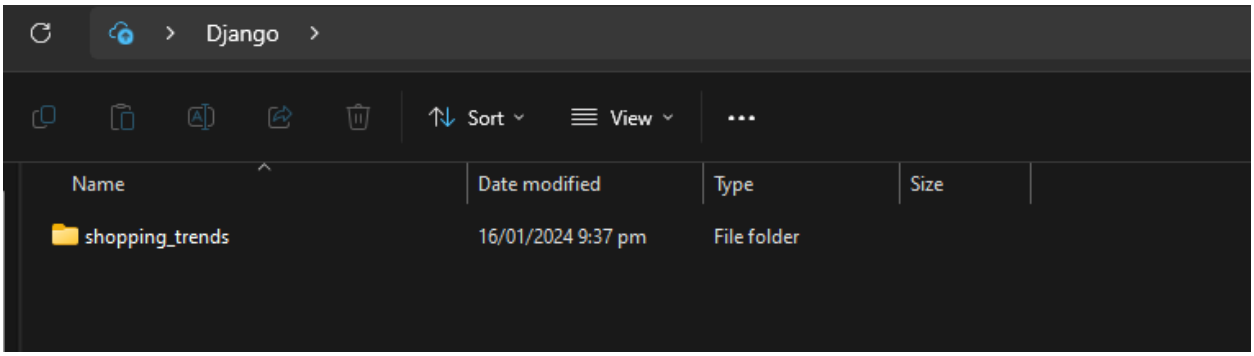


CREATING PROJECT

Create a folder on desktop named Django

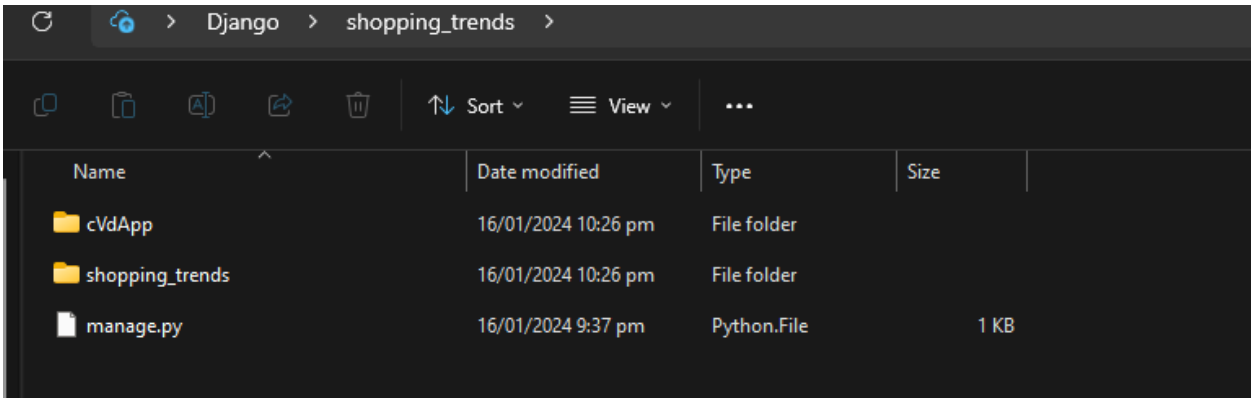
Go in terminal and type the following in VSCODE

```
PS C:\Users\Laraib Ijaz\Desktop> cd Django
PS C:\Users\Laraib Ijaz\Desktop\Django> django-admin startproject shopping_trends
PS C:\Users\Laraib Ijaz\Desktop\Django> dir


    Directory: C:\Users\Laraib Ijaz\Desktop\Django
```



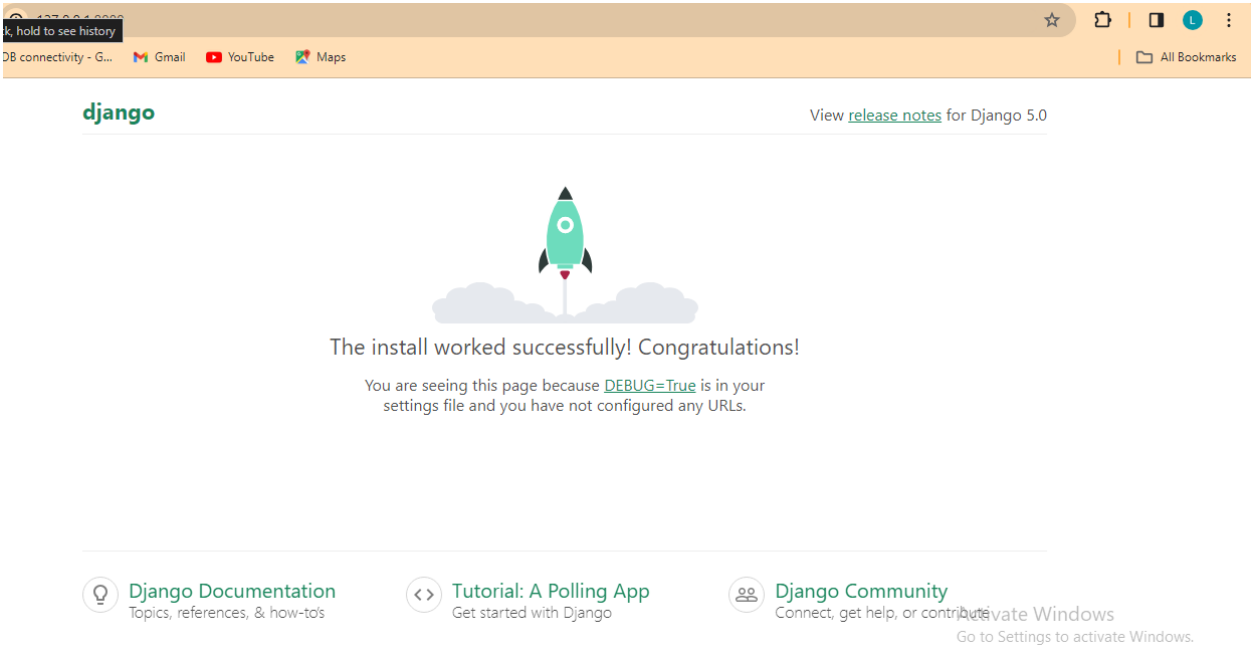
CREATING APP INSIDE DJANGO PROJECT

```
PS C:\Users\Laraib Ijaz\Desktop\Django> cd  shopping_trends
PS C:\Users\Laraib Ijaz\Desktop\Django\shopping_trends> python manage.py startapp cVdApp
PS C:\Users\Laraib Ijaz\Desktop\Django\shopping_trends>
```



RUN DJANGO SERVER

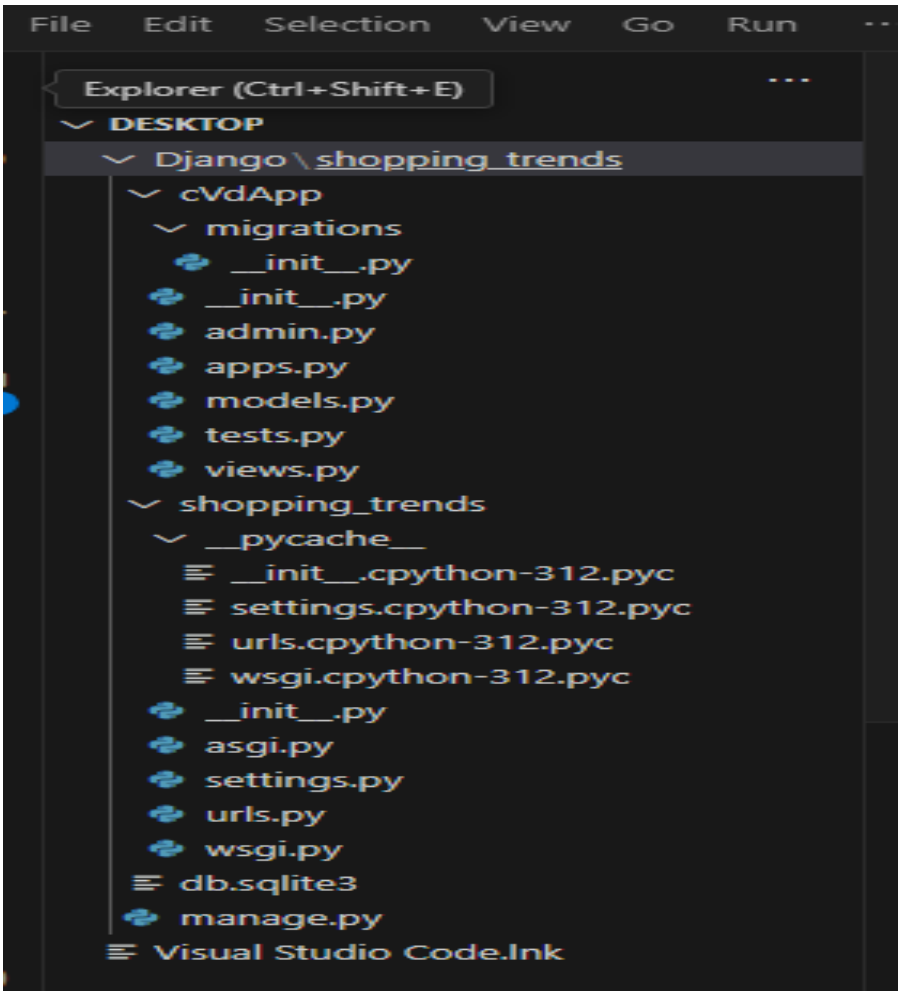
```
PS C:\Users\Laraib Ijaz\Desktop\Django\shopping_trends> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for
ontenttypes, sessions.
Run 'python manage.py migrate' to apply them.
January 16, 2024 - 22:32:53
Django version 5.0.1, using settings 'shopping_trends.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

[16/Jan/2024 22:35:17] "GET / HTTP/1.1" 200 10629
Not Found: /favicon.ico
[16/Jan/2024 22:35:23] "GET /favicon.ico HTTP/1.1" 404 2119
```



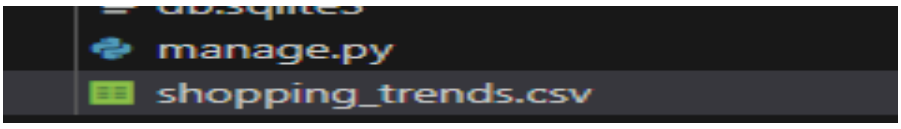
INSTALL VIRTUAL ENVIRONMENT

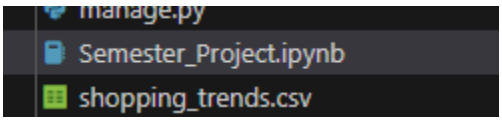
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Laraib Ijaz\Desktop\Django> pip install virtualenv
Collecting virtualenv
  Obtaining dependency information for virtualenv from https://files.pythonhosted.org/packages/83/22/54b1
437c3034c4bff92129c3e1e633079e2c4/virtualenv-20.25.0-py3-none-any.whl.metadata
  Downloading virtualenv-20.25.0-py3-none-any.whl.metadata (4.5 kB)
Collecting distlib<1,>=0.3.7 (from virtualenv)
  Obtaining dependency information for distlib<1,>=0.3.7 from https://files.pythonhosted.org/packages/8e/
a0b66367734e8faf3ec84bc0d412d8cfabbb66cd/distlib-0.3.8-py2.py3-none-any.whl.metadata
  Downloading distlib-0.3.8-py2.py3-none-any.whl.metadata (5.1 kB)
Collecting filelock<4,>=3.12.2 (from virtualenv)
  Obtaining dependency information for filelock<4,>=3.12.2 from https://files.pythonhosted.org/packages/8
dee7b59a8d4970eccdd44b99fe728ed912106fc781/filelock-3.13.1-py3-none-any.whl.metadata
  Downloading filelock-3.13.1-py3-none-any.whl.metadata (2.8 kB)
Collecting platformdirs<5,>=3.9.1 (from virtualenv)
  Obtaining dependency information for platformdirs<5,>=3.9.1 from https://files.pythonhosted.org/package
1a76d0043e018172db324a23fcdac70f77a551c11f618/platformdirs-4.1.0-py3-none-any.whl.metadata
  Downloading platformdirs-4.1.0-py3-none-any.whl.metadata (11 kB)
Downloading virtualenv-20.25.0-py3-none-any.whl (3.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.8/3.8 MB 639.6 kB/s eta 0:00:00
Downloading distlib-0.3.8-py2.py3-none-any.whl (468 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 468.9/468.9 kB 814.6 kB/s eta 0:00:00
Downloading filelock-3.13.1-py3-none-any.whl (11 kB)
Downloading platformdirs-4.1.0-py3-none-any.whl (17 kB)
Installing collected packages: distlib, platformdirs, filelock, virtualenv
Successfully installed distlib-0.3.8 filelock-3.13.1 platformdirs-4.1.0 virtualenv-20.25.0

[notice] A new release of pip is available: 23.2.1 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
PS C:\Users\Laraib Ijaz\Desktop\Django>
```

OPEN THE DJANGO FOLDER FROM FILE OPEN PRESENT

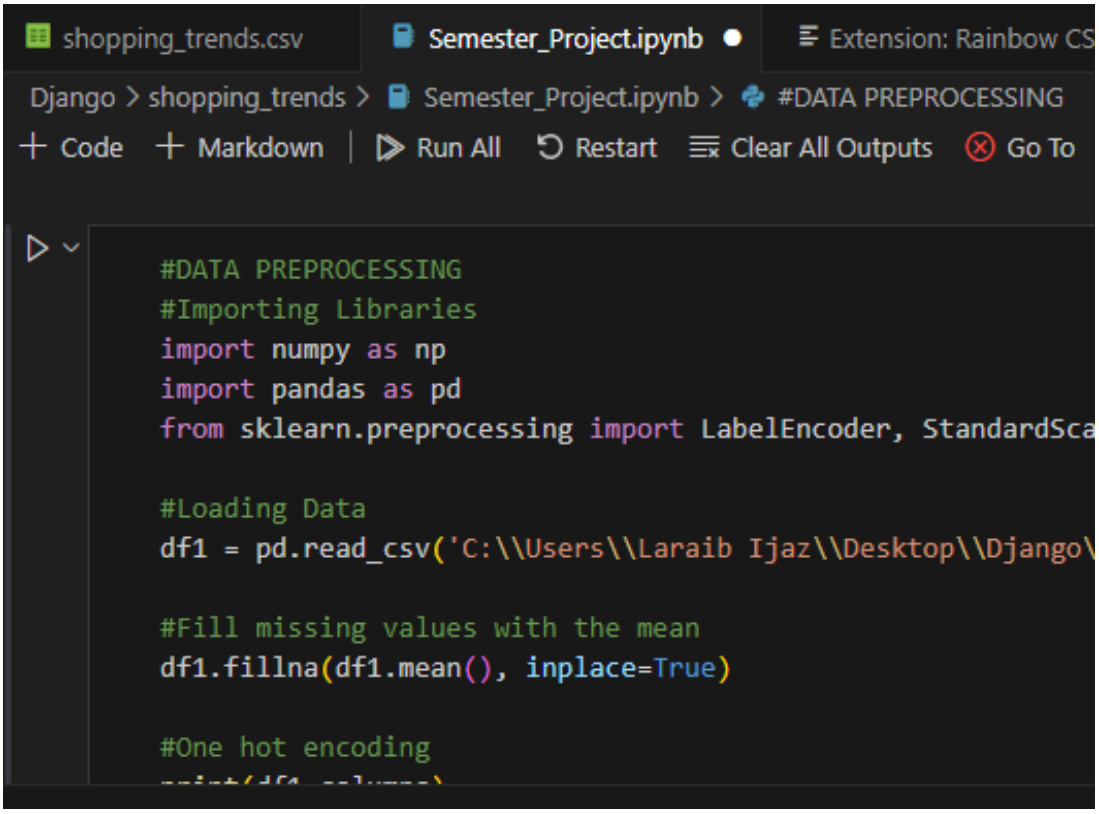


Make a new file with csv extension and add the csv data by copying it from that file





ADD ENTIRE CODE IN IPYNB FILE



1.numpy

```
PS C:\Users\Laraib Ijaz\Desktop\Django> cd shopping_trends
PS C:\Users\Laraib Ijaz\Desktop\Django\shopping_trends> pip install numpy
Collecting numpy
  Obtaining dependency information for numpy from https://files.pythonhosted.org/packages/ad/11/
ed6d3519afb14f9771fac1c9b7de/numpy-1.26.3-cp312-cp312-win_amd64.whl.metadata
  Downloading numpy-1.26.3-cp312-cp312-win_amd64.whl.metadata (61 kB)
     ---------------------------------------- 61.2/61.2 kB 98.8 kB/s eta 0:00:00
Downloading numpy-1.26.3-cp312-cp312-win_amd64.whl (15.5 MB)
     ---------------------------------------- 15.5/15.5 MB 985.4 kB/s eta 0:00:00
Installing collected packages: numpy
Successfully installed numpy-1.26.3
```

2.pandas

```
PS C:\Users\Laraib Ijaz\Desktop\Django\shopping_trends> pip install pandas
Collecting pandas
  Obtaining dependency information for pandas from https://files.pythonhosted.org/packages/ae/d9/3741b34
89992ad9962196a62721ef9980045/pandas-2.1.4-cp312-cp312-win_amd64.whl.metadata
  Downloading pandas-2.1.4-cp312-cp312-win_amd64.whl.metadata (18 kB)
Requirement already satisfied: numpy<2,>=1.26.0 in c:\users\laraib ijaz\appdata\local\programs\python\py
s (from pandas) (1.26.3)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\laraib ijaz\appdata\roaming\python\pyt
om pandas) (2.8.2)
Collecting pytz>=2020.1 (from pandas)
  Obtaining dependency information for pytz>=2020.1 from https://files.pythonhosted.org/packages/32/4d/a
a8119f00d74ae9c2efa79f8ef9994261fc2/pytz-2023.3.post1-py2.py3-none-any.whl.metadata
  Downloading pytz-2023.3.post1-py2.py3-none-any.whl.metadata (22 kB)
Requirement already satisfied: tzdata>=2022.1 in c:\users\laraib ijaz\appdata\local\programs\python\pyth
(from pandas) (2023.4)
```

3.skykit-learn

```
PS C:\Users\Laraib Ijaz\Desktop\Django\shopping_trends> pip install scikit-learn
Collecting scikit-learn
  Obtaining dependency information for scikit-learn from https://files.pythonhosted.org/packages/fe/6b/db949e
0f340b7715d22f0c9c965bdf07de6ca75a3/scikit_learn-1.3.2-cp312-cp312-win_amd64.whl.metadata
  Downloading scikit_learn-1.3.2-cp312-cp312-win_amd64.whl.metadata (11 kB)
Requirement already satisfied: numpy<2.0,>=1.17.3 in c:\users\laraib ijaz\appdata\local\programs\python\pytho
ges (from scikit-learn) (1.26.3)
Collecting scipy>=1.5.0 (from scikit-learn)
  Obtaining dependency information for scipy>=1.5.0 from https://files.pythonhosted.org/packages/c6/a1/357e4c
e9a5ea8aaaa6b702833c81be11670dcbad8/scipy-1.11.4-cp312-cp312-win_amd64.whl.metadata
  Downloading scipy-1.11.4-cp312-cp312-win_amd64.whl.metadata (60 kB)
     ---------------------------------------- 60.4/60.4 kB 133.8 kB/s eta 0:00:00
Collecting joblib>=1.1.1 (from scikit-learn)
  Obtaining dependency information for joblib>=1.1.1 from https://files.pythonhosted.org/packages/10/40/d5511
```

4. seaborn

```
PS C:\Users\Laraib Ijaz\Desktop\Django\shopping_trends> pip install seaborn
Collecting seaborn
  Obtaining dependency information for seaborn from https://files.pythonhosted.org/packages/2d/46/cf3fce
59d591df446ec716e72c3b4ce71b34/seaborn-0.13.1-py3-none-any.whl.metadata
  Downloading seaborn-0.13.1-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in c:\users\laraib ijaz\appdata\local\programs\pytho
kages (from seaborn) (1.26.3)
Requirement already satisfied: pandas>=1.2 in c:\users\laraib ijaz\appdata\local\programs\python\python3
om seaborn) (2.1.4)
Collecting matplotlib!=3.6.1,>=3.4 (from seaborn)
  Obtaining dependency information for matplotlib!=3.6.1,>=3.4 from https://files.pythonhosted.org/packa
d6fb440e811b754fc3950d6868c38ace57d0632b674415/matplotlib-3.8.2-cp312-cp312-win_amd64.whl.metadata
  Downloading matplotlib-3.8.2-cp312-cp312-win_amd64.whl.metadata (5.9 kB)
```